Virtuoso Universal Server 4.5 Application Integration (Web Service Composition) Reviewer's Guide

OpenLink Software
Published: March 9, 2006

Table of Contents

1 Preface

1.1 Objective

Thank you for taking the time to evaluate OpenLinks Virtuoso Universal Server 4.5. This guide introduces the Virtuoso 4.5 Server as well as Virtuoso's new administration and configuration tool, the Virtuoso Conductor.

This guide takes you through Virtuoso, covering virtual and federated database, publishing and utilizing web services, XML storage, processing and querying, business process management and more.

We illustrate these aspects of the product with sample code and Queries against the Virtuoso Demo database as well as links to documentation, tutorials and other key resources.

OpenLink Software hopes you find this guide informative while you learn what Virtuoso has to offer for your enterprise.

For More information on Virtuoso or OpenLink Software, contact us at:

Sales:	sales@openlinksw.com
General	general.information@openlinksw.com
Technical Support	technical.support@openlinksw.com

Or visit our web sites:

OpenLink Software	http://www.openlinksw.com
Virtuoso	http://virtuoso.openlinksw.com

1.2 System Requirements

The Virtuoso Universal Server installation contains the Virtuoso VDBMS, server administrator web files, and Virtuoso client connectivity.

Software

- Virtuoso Universal Server 4.5
- Virtuoso Conductor VAD 1.00.3051

Virtuoso is available on a number of hardware platforms. Listed below are some of the operating system ports currently available.

Hardware

- Microsoft Windows 2000, XP (32 and 64 bit platforms), Server 2003
- Linux (Intel and AMD64) (glibc2.1, glibc2.2, glibc2.3 compatible)
- Linux (IBM pSeries 64 bit)
- Mac OS X 10.2+ ("Jaguar") & 10.3+ ("Panther")
- Solaris 2.8 / 2.9 (Sparc)
- AIX 4.3.x/5.0.x/5.1.x 32 and 64 bit Power PC

Operating Systems and Browsers

- IE 6.0, Mozilla (Firefox), Opera, etc.

1.3 Installation

Download the Virtuoso 4.5 installer from OpenLink Software Virtuoso Download area at: http://oplweb.openlinksw.com:8080/download. Install both default and demo databases by following the instructions provided by the installer. At the end of the installation process, the Virtuoso server will be installed with both default and demo databases.

On Windows, the Virtuoso start menu will appear which provides menu options for the Conductor, Web Applications, the 3.5 Classic Administration tool and links to other resources as shown in **Figure 1 - Virtuoso Start Menu**.

Figure 1 - Virtuoso Start Menu

2 Virtuoso Universal Server

2.1 Introduction

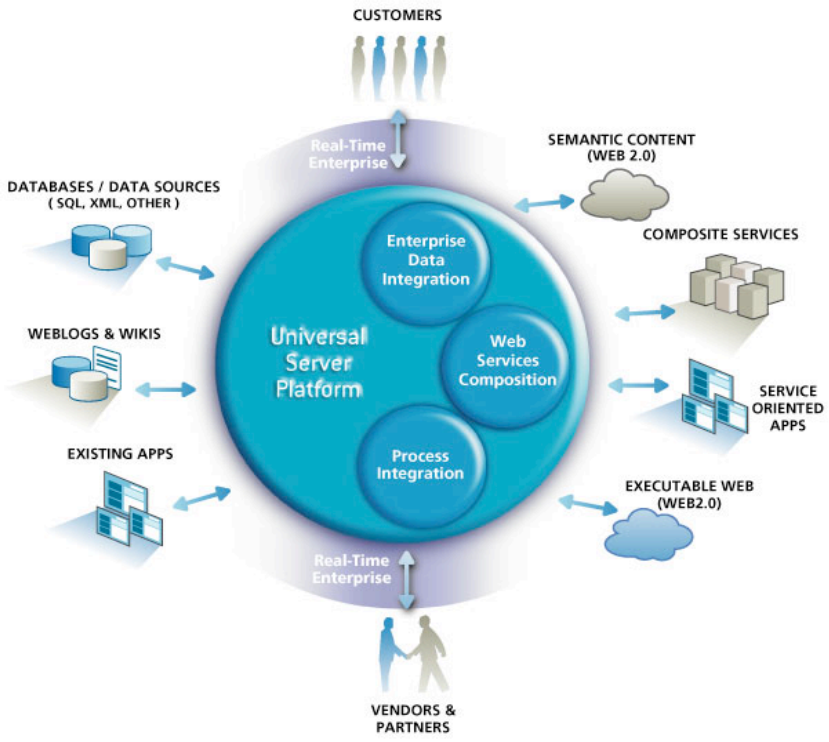
This reviewer's guide presents Virtuoso 4.5, the latest version of OpenLink Software's next-generation Universal Server Platform. Virtuoso facilitates the development and deployment of a new generation of Enterprise-wide, Internet-, Intranet-, and Extranet-based solutions. Virtuoso capabilities include handling of Disparate Databases & Data Sources, Web Service Composition and Business Process Integration**.**

The conceptual architecture diagram below highlights Virtuoso's key features.

Figure 2 - Virtuoso Conceptual Architecture including:

- **Enterprise Data Integration** –*Virtuoso Virtual Database* Engine
- **Web Services Composition –***Web Services Platform*
- **Introduction to Process Integration –** *Blog, BPEL, etc.*

Figure 2 - Virtuoso Conceptual Architecture



The sections throughout this guide are aimed at demonstrating Virtuoso's value proposition in these key areas. The step-by-step examples will quickly introduce you to Virtuoso and show you many useful techniques and features. The examples grouped by area include:

Enterprise Data Integration

- Identifying disparate data for integration and linking the data into Virtuoso
- Creating XML based documents or ODBC data sources for heterogeneous data

Web Services Composition

- Identifying monolithic applications from which services are to be decoupled, composing, and linking their functionality into Virtuoso
- Creating virtual directories as endpoints for exposing functionality for web services clients.

Process Integration

- See the OpenLink BPEL Process Manager Reviewers Guide for more information.

3 Web Services Platform

3.1 What Is This?

As a Web Services Platform, Virtuoso includes built-in support for WSDL, SOAP, UDDI WS-Security, and several other Web Services Protocols.

It enables the creation and composition of Web Services from existing application logic pools hosted within monolithic mission critical applications. Thus, the process of creating a SOAP and WSDL compliant Web Service from existing application logic resolves to the following steps:

- Identify the SQL Stored Procedures, Java Classes, .NET Assemblies, or C/C++ modules implementing the functionality in question.
- Use the HTML UI to trigger the generation of a WSDL file and SOAP invocation wrappers for the application logic
- Use Virtuoso's HTTP Server functionality to create a Virtual Directory that acts as an execution endpoint for your Web Service
- Test the usability of the new Web Service by interacting directly with an HTML based Web Service verification page
- Proceed to using your newly created service with a Web Service aware development tool, environment, or service
- Register your newly created service with Virtuoso's in-built UDDI Server, and then
- advertise your service internally or externally to other service consumers.

Virtuoso facilitates the creation of SOAP compliant Web Services from existing or legacy monolithic applications without locking you into a host operating system, programming language/environment, or database engine. It enables you to perform the prerequisite step of creating and/or composing Web Services as part of your effort to cost effectively exploit emerging application architecture principles such as Service Oriented and Event Driven Architectures.

3.2 The Value Proposition

Service Composition by exposure of existing time-tested application logic for invocation using Web Services protocols without any code re-writes. Code format support includes SQL Stored Procedures, .NET assemblies, Java Classes, C/C++ modules, etc.

Service Invocation Endpoints via HTTP/WebDAV based virtual directory and multi-homing functionality that provides endpoints for SOAP-, WS-Security-, WSDL-, and UDDI-compliant interactions with composite services.

3.3 Exploiting Value Proposition

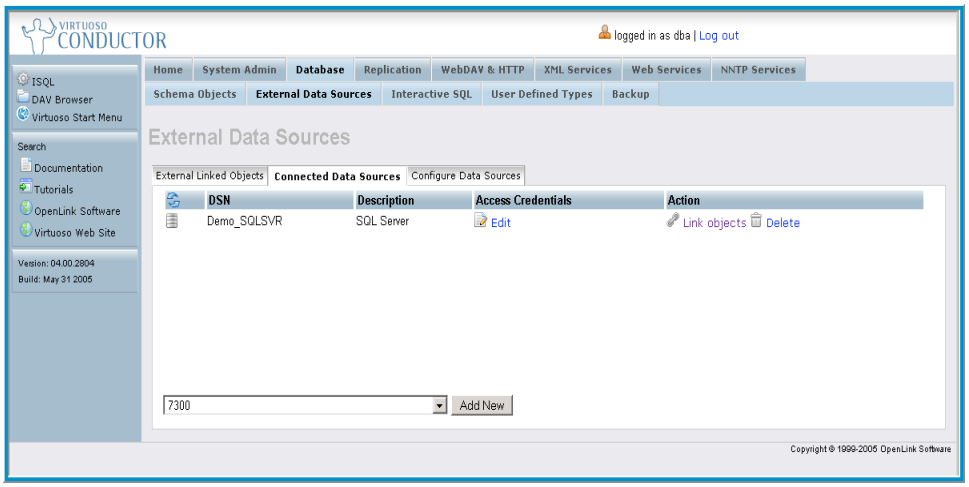
3.3.1 Web Service Composition from SQL Stored Procedures

3.3.1.1 Linking/Attaching External SQL Stored Procedures

The following section includes steps for linking in external stored procedures from an ODBC Datasource. The steps in the example below uses a connected datasource called Demo_SQLSVR that maps to a Microsoft SQL Server database containing the Northwind schema.

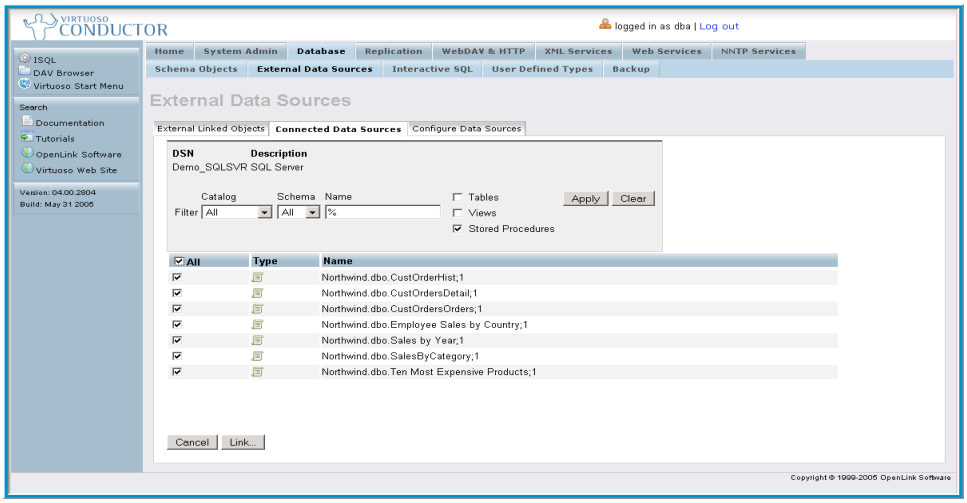
Step 1. From the Conductor, navigate to the Database Tab, select the External **Datasources sub-tab** and then the Connected Data Sources tab Select the Link objects link under the Actions column as shown below in **Figure 3 – External Procedures Management.**

Figure 3 – External Procedures Management



Step 4. As seen in **Figure 4 - Selecting the External Stored Procedures** below, when the page returns, a list is displayed of the available remote procedures. You can link any of the listed procedures into Virtuoso by selecting them from the list and pressing the Link Selected button.

Figure 4 - Selecting the External Stored Procedures



Step 5. In **Figure 5 – Linking and Qualifying the Stored Procedures**, a new page listing the chosen procedures and their associated data type information is displayed. On this form, you can alter the data type mappings that Virtuoso will use both internally and for any future interactions with the SOAP server. If you do not want to specify any special type information, the details can be left as default.

Figure 5 – Linking and Qualifying the Stored Procedures

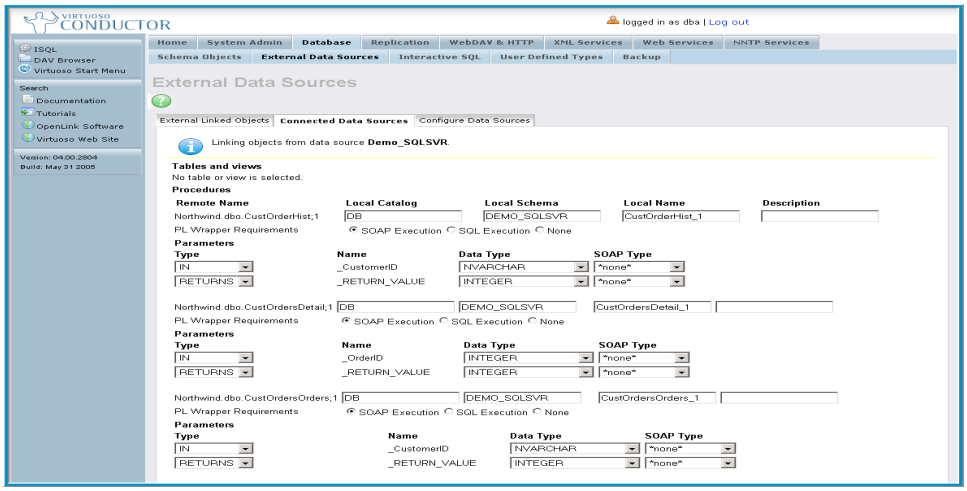
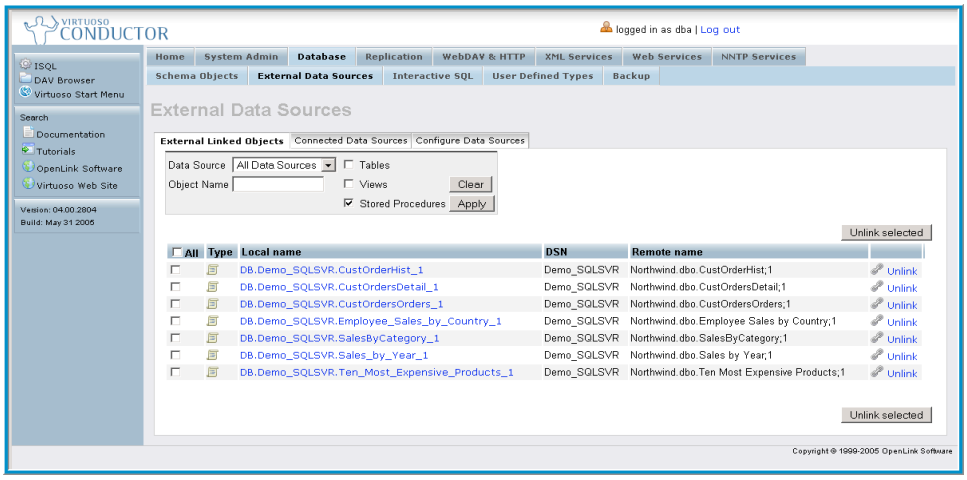


Figure 6 – External Data Sources



Note 1 - When linking in these procedures, for each remote procedure you have the option to change how they will be referenced within Virtuoso. These include changing the fields for** Local Name**, Database, and** Schema**.

Step 6. As shown in **Figure 6 – External Data Sources** above, for each procedure there are radio buttons for selecting the **PL Wrapper Requirements**. This option is of particular importance for remote procedures capable of returning a result set. Remote procedures can be linked using a Virtuoso PL wrapper meaning that Virtuoso procedure language code provides a mechanism for negotiating the result set. The available options are:

- **Soap Execution** - Generates a PL wrapper that can be published to Web Services.
- **SQL Execution** - Generates a PL Wrapper that is more suitable for general SQL use.
- **None** - Does not generate and PL Wrapper code and simply links the procedure by reference.

3.3.1.2 Publishing SQL Stored Procedures as Web Services

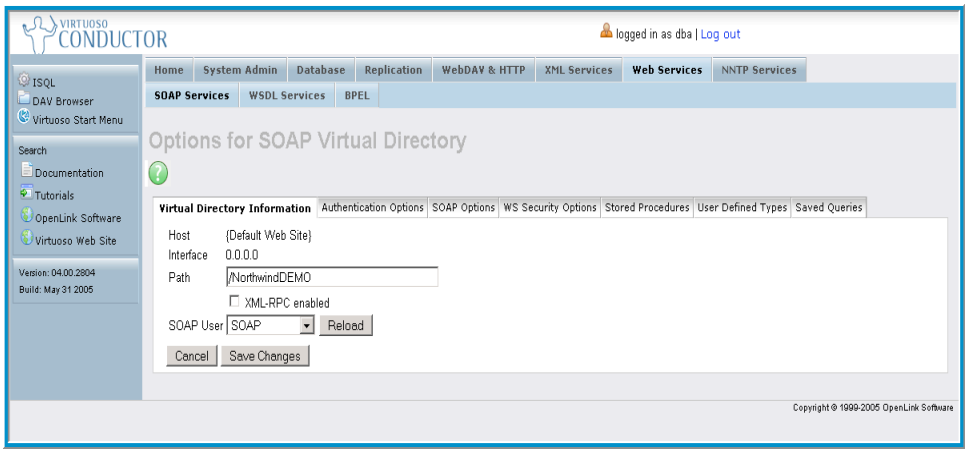
Before any procedures native or linked in exposed as SOAP Web Service a location in HTTP space must be defined. As mentioned in the previous section, the

/WebDAV/HTTP Virtual Directories page lets you make a new URL Mappings. The HTTP Virtual Directories page should contain a table with a row for {Default Web Site}. Click on the **Edit URL Mappings** link for the {Default Web Site} line to begin defining a new SOAP mapping.

Step 7. To create a Virtual Directory with a SOAP mapping you first must select the ***Soap Service HTTP endpoints*** option under the Web Services Tab in the Conductor. Expand the folder on the right and click the Add New Directory button. Select the options for the SOAP Virtual Directory and Click the NEXT button.

Step 8. Once selected, as seen in **Figure 7 – Creating a Virtual Directory using the SOAP Template** below, a list of existing mappings is shown. Press the **ADD VIRTUAL DIRECTORY** button to create a new mapping. A list of templates is then displayed. However, instead of selecting File; select the SOAP template.

Figure 7 – Creating a Virtual Directory using the SOAP Template



Step 4. In ***Figure 96***, you are presented with the main configuration page for adding a Virtual Web Directory. Particular options to note are Listener Details and SOAP Options.

Figure 8 – Adding a Virtual Web Directory to Publish the Service

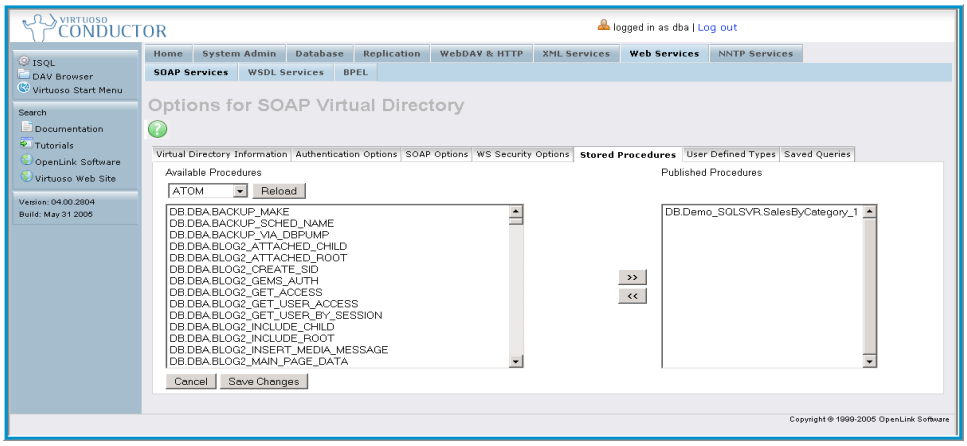
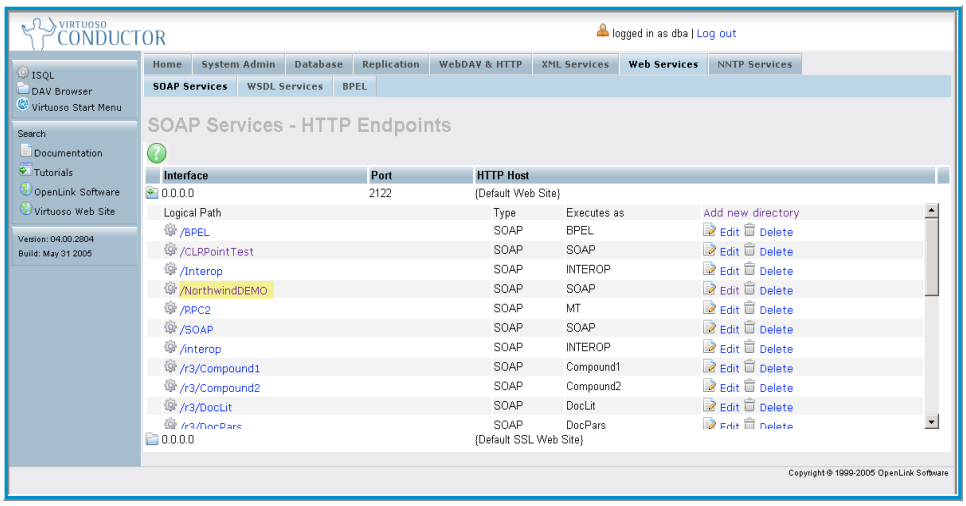


Figure 9 – Publishing the SQL Stored Procedure



You will be presented with a screen for selecting the procedures or XML templates. For selecting the procedures, you first have to select the appropriate catalogue that contains the procedure that you want to publish. For the XML Templates you simply select the template using the checkbox and provide an **Export As** name to choose the new SOAP name that a XML template for which it is referred.

Figure 10 – Web Services Test Page

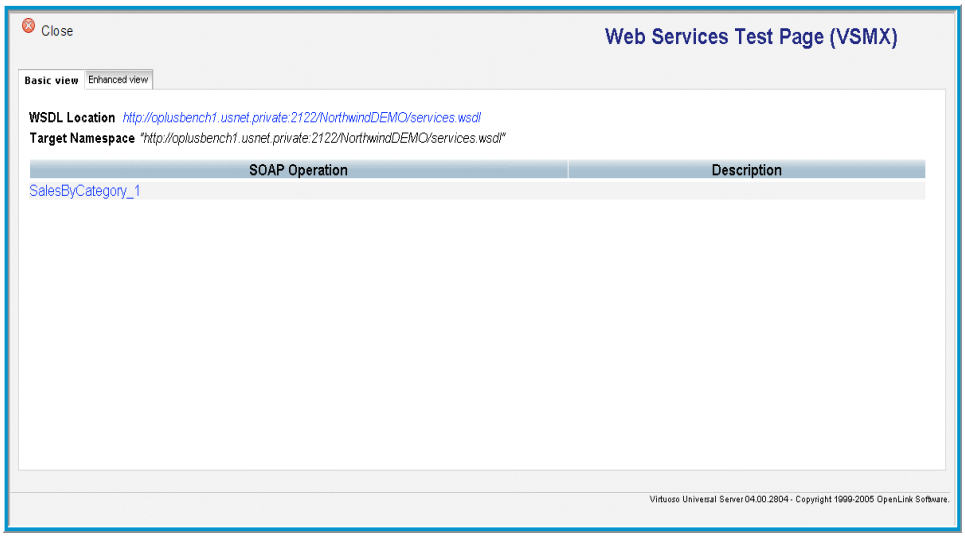


Figure 11 – Selecting the Database containing the stored procedure

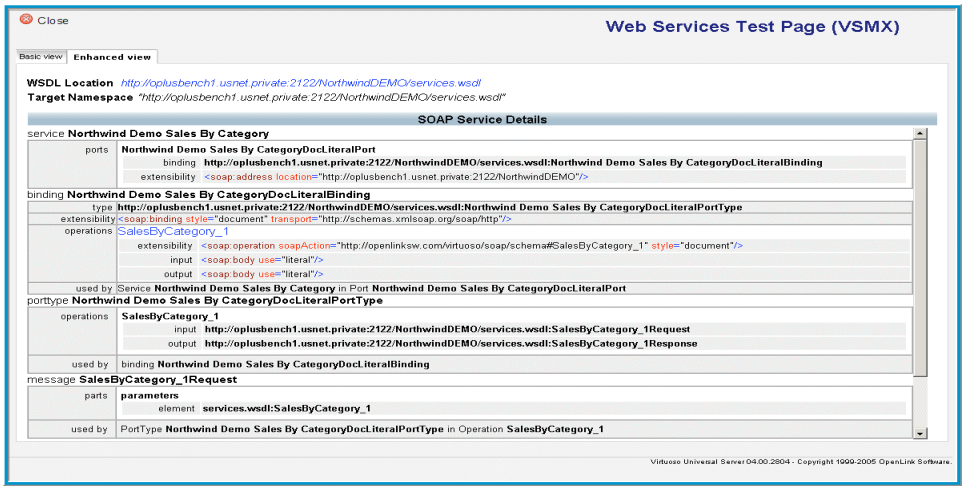


Figure 12 - Web Services Test Page Service Invocation Response

More Information

For more information regarding Publishing 3rd Party stored procedures as Web Services see

- Virtuoso Publishing 3rd Party Stored Procedures as Web Services Documentation
- Virtuoso Publishing 3rd Party Stored Procedures as SOAP Services Tutorial

3.3.1.3 Exercising Published Services

3.3.2 Web Services Composition from other Application Logic Sources

Business logic written in Java or any of the Microsoft .Net languages can be hosted inside the Virtuoso process. This business logic can access Virtuoso's database and virtual database capabilities directly via an in-process client. This eliminates the gap between the application and database tiers in a three-tier architecture.

If structures of the hosted language are to be transferred, they will first be mapped to a SQL type, which then declares the precise SOAP types for the members

User Defined Types

To Virtuoso SQL, instances of hosted classes appear just as native SQL user-defined type instances or types (UDTs). Their methods can be called, their members can be accessed and the instances may be stored into columns of relational tables.

These user-definable datatypes can be based on any hosted language or class and can be used to define database table columns.

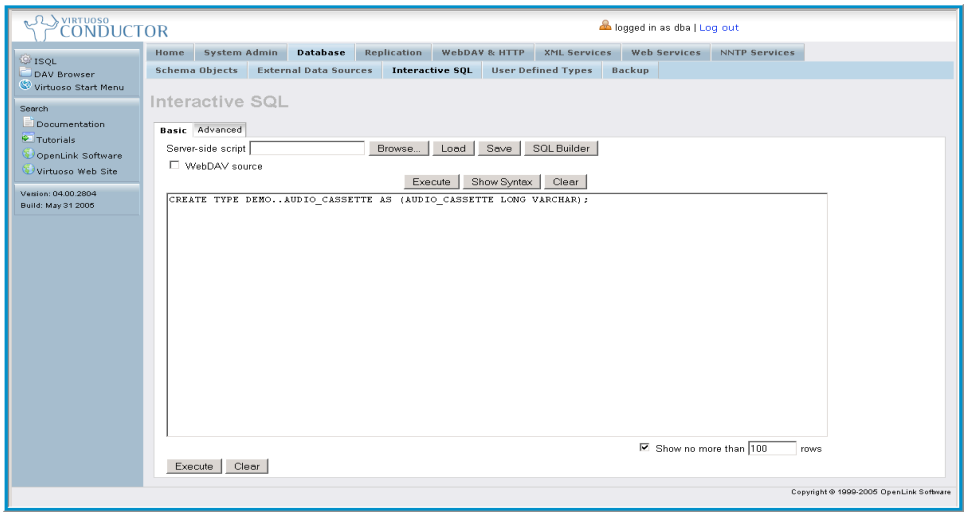
The following code fragment shows how to define a SQL user-defined type, which shows the basic syntax used for the ***CREATE TYPE***. Refer to the User Defined Types and Create Type Syntax CREATE TYPE in the Virtuoso Guide for complete details on syntax.

Step 1. Using ISQL as shown in **Figure 13 – SQL Create Type Syntax**, Type in the following syntax for the User Defined Type and click the Execute button.

Note 2 - By default if you do not specify the language the create type automatically defaults to SQL.

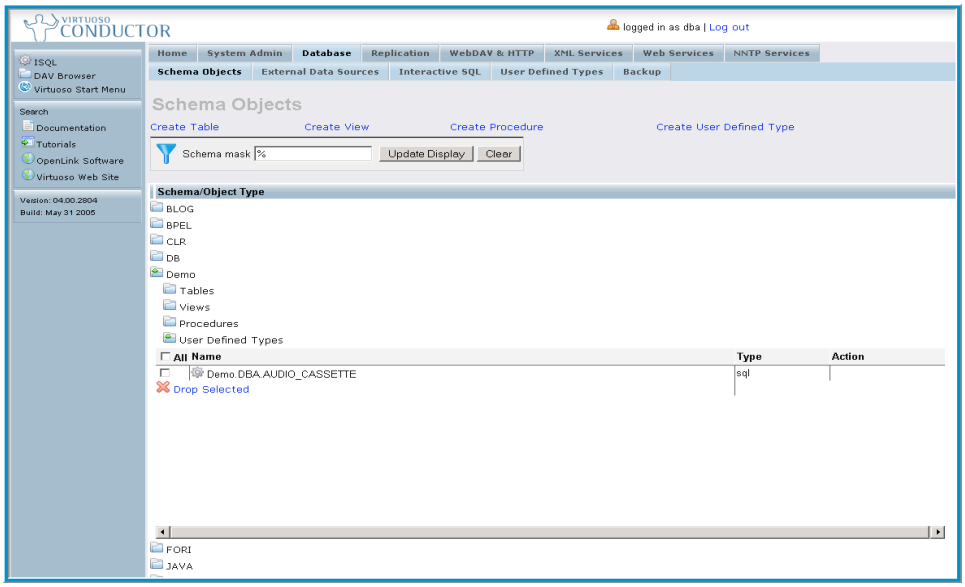
Figure 13 – SQL Create Type Syntax

```
CREATE TYPE DEMO..AUDIO_CASSETTE AS (AUDIO_CASSETTE LONG VARCHAR);
```



Step 2. One the UDT has been defined, verify the UDT by navigating to the Database TAB, selecting Schema Objects and expanding the Demo database and selecting the User Defined Types as show below in **Figure 14 – Demo AUDIO_CASSETTE User** `Defined Type`.

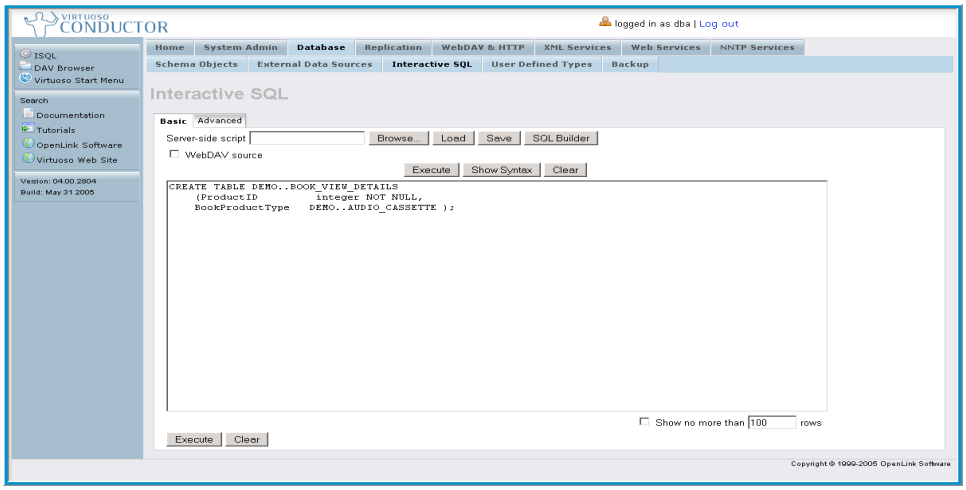
Figure 14 – Demo AUDIO_CASSETTE User Defined Type



Step 3. Once the UDT has been defined, it can then be used in stored procedures, tables or views. For example, in the create table below Instead of specifying ***BookProductType*** as the data type you can specify ***DEMO..AUDIO_CASSETTE*** as show below in **Figure 15 – Create Table Statement using UDT.**

Figure 15 – Create Table Statement using UDT

```
CREATE TABLE DEMO..BOOK_VIEW_DETAILS
    (ProductID          integer NOT NULL,
    BookProductType    DEMO..AUDIO_CASSETTE );
```



Step 4. Verify the table referencing the UDT by navigating to the Database TAB, selecting Schema Objects and expanding the Demo database and selecting the Tables as show below in **Figure 16 – Demo Table BOOK_VIEW_DETAILS.** Select the details of the table by clicking on the Definitions as shown in **Figure 17 – Schema Object BOOK_VIEW_DETAILS Definitions.**

Figure 16 – Demo Table BOOK_VIEW_DETAILS

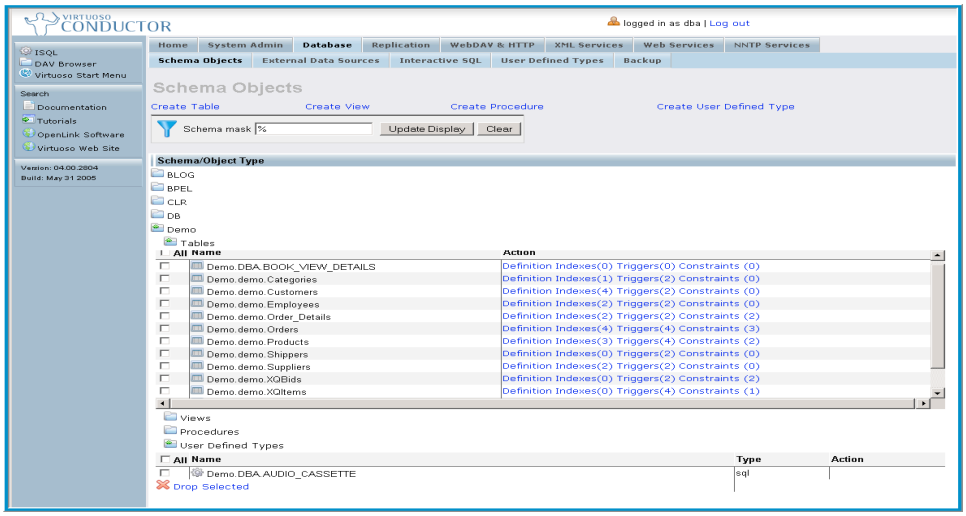
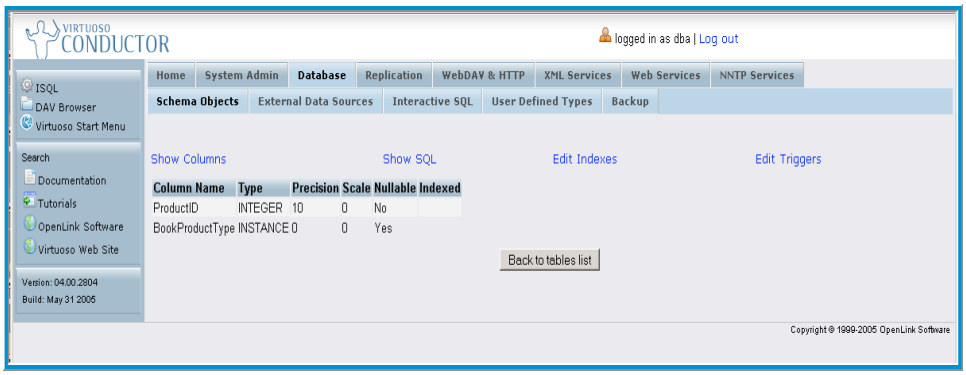


Figure 17 – Schema Object BOOK_VIEW_DETAILS Definitions



New types can be further derived producing sub-types. User-defined types can include methods and constructors to create any potentially complicated system to house data as exactly required. Virtuoso automatically handles the mapping between the SQL type system and hosted language native types.

Now that we have seen how to define types the examples below will show how they can be used to hosts a Java VM or C# and allow manipulation of classes through the SQL user defined types.

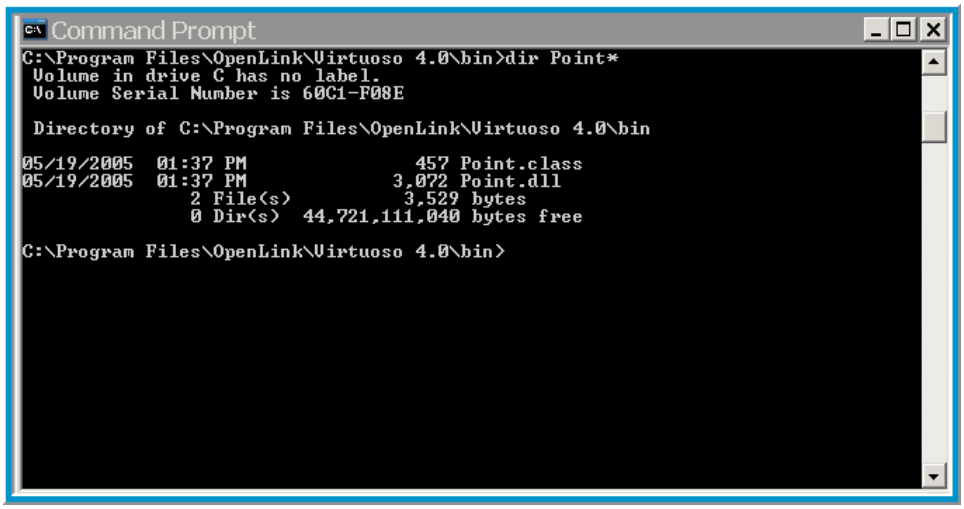
Note 3 - At install time, you must select the runtime hosting option for Java and the CLR on Windows and Java and Mono/CLR as the interfaces require Mono/CLR and Java extension servers. In addition, All Java classes have to be in the CLASSPATH of the hosted Java VM.

3.3.2.1 Runtime Hosting using C# and Java objects to extend Virtuoso via User Defined Types

3.3.2.1.1 *User Defined Types and CLR*

The examples in this section show hosting a distance calculation written C# hosted by Virtuoso. In the steps, we will define a class ***point*** written in both Java and C#, which can be found in the bin directory under Virtuoso 4.5 which are automatically installed. For example on Windows C:/Program Files/OpenLink/Virtuoso 4.5/bin.

Figure 18 – Directory of Point.dll and Point.class



Step 1. The first step is to create a piece of code or class in C#. ***Figure 111*** *sh*ows the definition of the point class written in C#.

Figure 19 - C# code Point.cs

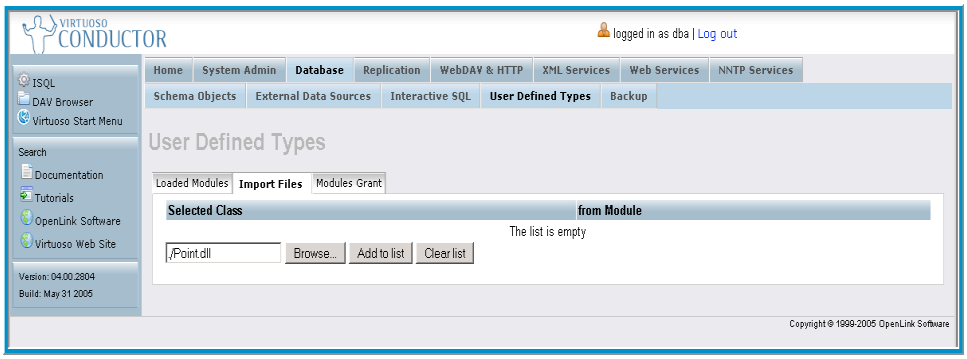
```
using System;

[Serializable]
public class Point
{
  public Double x;
  public Double y;

  public Point_10 ()
    {
      x = 0;
      y = 0;
    }
  public Point_10 (Double new_x, Double new_y)
    {
      x = new_x;
y = new_y;
}
public Double distance (Point p)
  {
Double ret;
       ret =  Math.Sqrt ((p.x - this.x) * (p.x - this.x) + (p.y -
this.y) * (p.y - this.y));
return ret;
  }
}
```

Step 2. The next step is to register the "Point" Class with Virtuoso as a UDT by first selecting the menu tree of the Conductor and select Runtime Hosting. Then select the Import Files option as show below in **Figure 20 – Import Files**.

Figure 20 – Import Files



Step 3. Type in the location and file name for the class. In this example, we type in C:/Program Files/OpenLink/Virtuoso 4.5/bin/Point.dll.

Note 4 - Under this option, you can load as many modules as you want to load by selecting Add to List and then select Load all.

Step 4. Once you have added the object, select ***Load Selected***, which will register the Point class object as a UDT shown below in **Figure 21 – Loading Modules** and **Figure 22 – Loading Hosted Modules**.

Figure 21 – Loading Modules

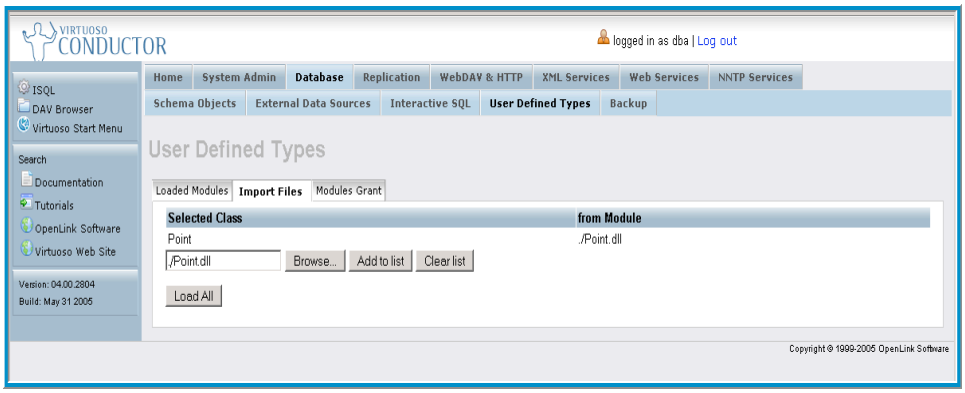
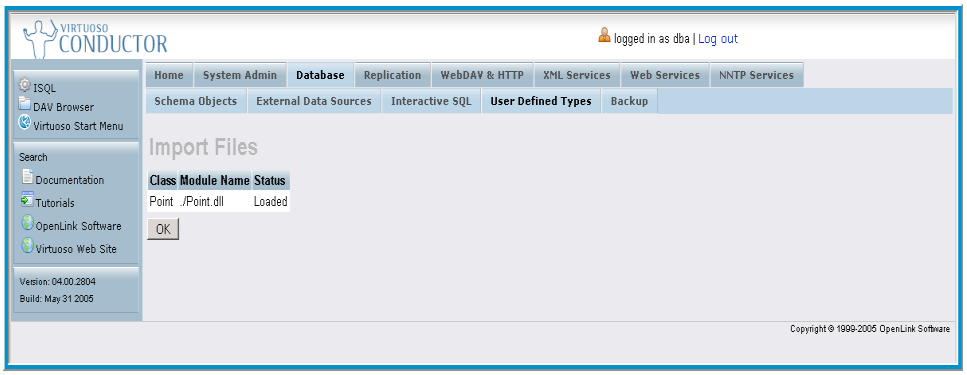


Figure 22 – Loading Hosted Modules



Step 5. In **Figure 23 – Applying Module Grants**, we add the appropriate Grants to the class for execution and access. In case will are applying the least

restrictive by granting the module public access. The Security of a UDT is maintained through normal SQL GRANT and REVOKE statements via a simple extension. You can define the level of access to both native and externally hosted UDTs.

Step 6. Once grants are applied, you can view the existing grants for all loaded modules as shown in **Figure 24 – Existing Grants**. These grants can be removed at anytime and redefined using this screen. This also can be grants can also be applied via the SQL statement GRANT/REVOKE EXECUTE on <user_defined_type>:

Example: Grant Public on CLR..Point;

Figure 23 – Applying Module Grants

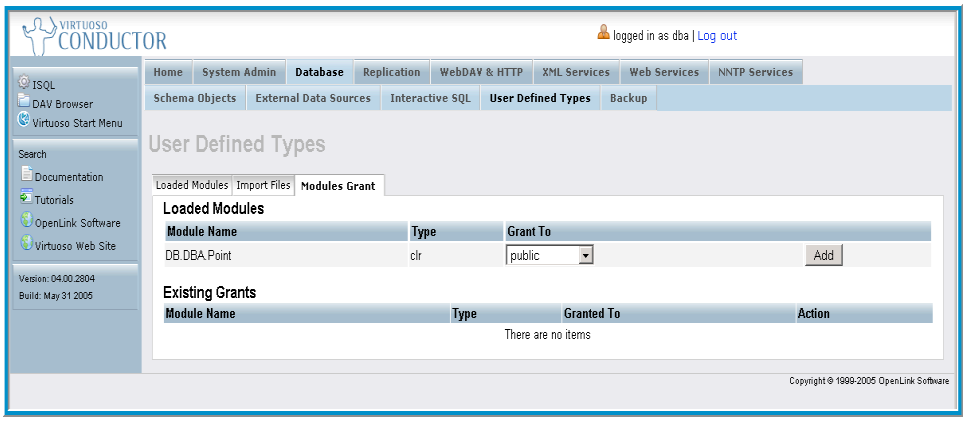
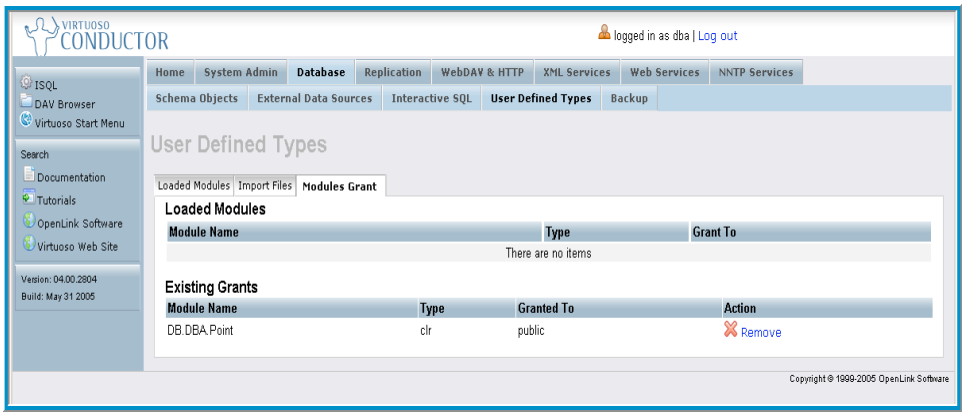


Figure 24 – Existing Grants



The equivalent Virtuoso Procedure Language statement for defining the classes with the import function call is shown below in **Figure 25 – SQL Equivalent Create Type in Virtuoso for CLR**.

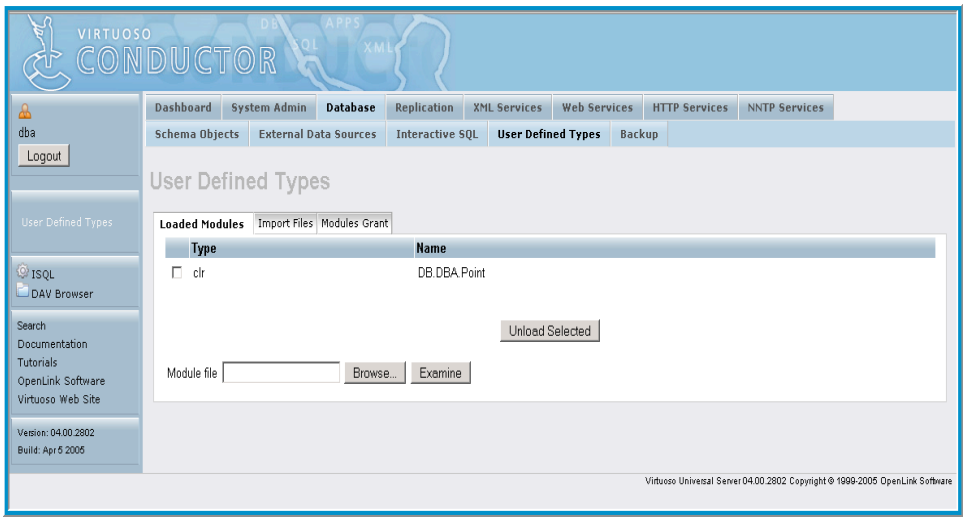
Figure 25 – SQL Equivalent Create Type in Virtuoso for CLR

```
create type "Point" language CLR external name 'Point/Point'
 AS (
     "x" double precision external name 'x' external type 'System.Double',
     "y" double precision external name 'y' external type 'System.Double')
 unrestricted
 CONSTRUCTOR METHOD "Point" (
    "new_x" double precision external type 'System.Double',
```

```
    "new_y" double precision external type 'System.Double'),
      METHOD "distance" ("p" Point external type 'Point')
    returns double precision external type 'System.Double'
    external name 'distance'
;
```

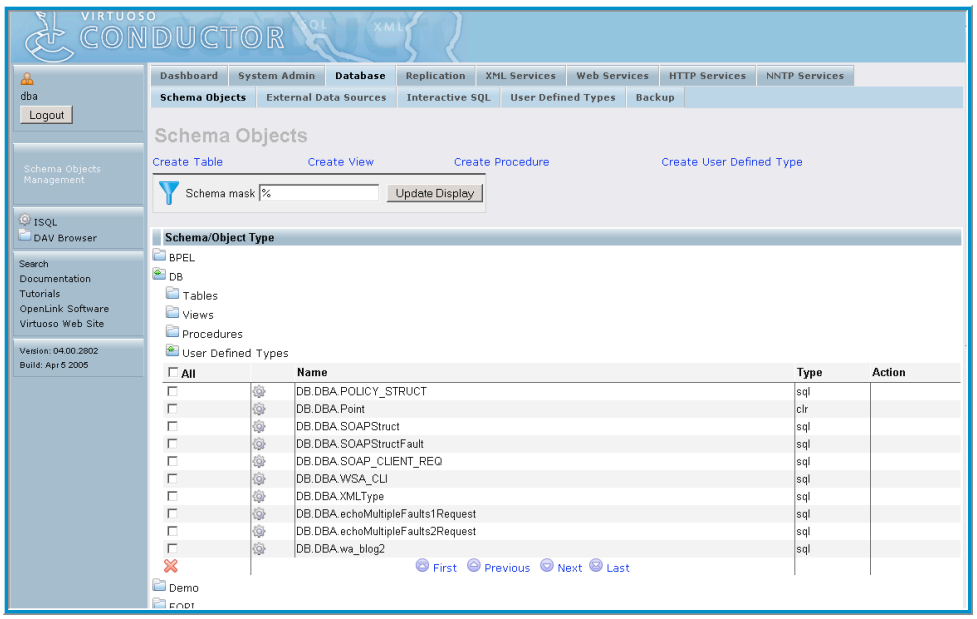
As seen in **Figure 26 – Viewing Hosted Module**, loaded Hosted Modules can be viewed at any time via the Administration Use Interface.

Figure 26 – Viewing Hosted Module



As show below in **Figure 27 – User Defined Types**, a list of user-defined types can be viewed by selecting the User Defined Types for the schema under the Database option in the Conductor.

Figure 27 – User Defined Types

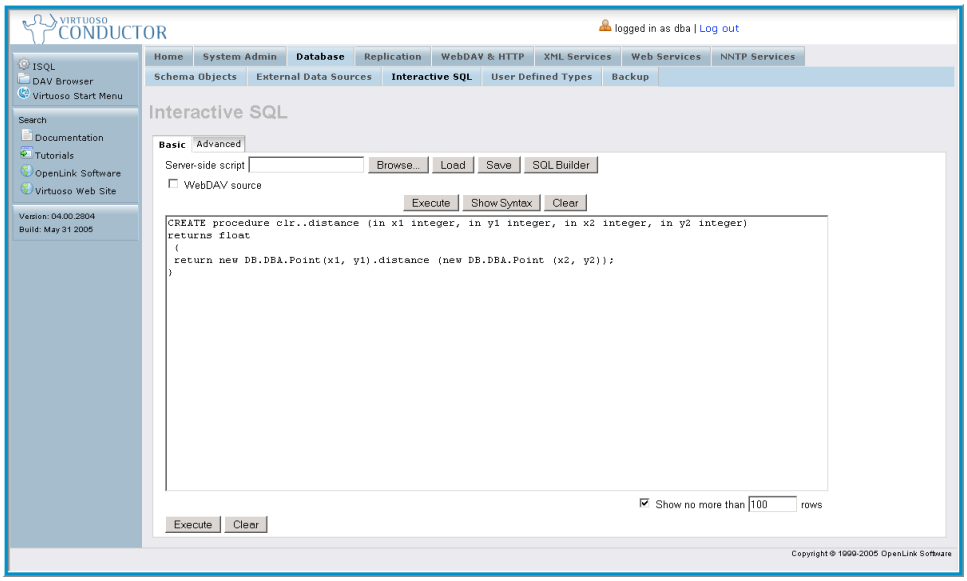


Step 7. To test this UDT we next define a stored procedure called distance, which can be executed via ISQL. The "distance" stored procedure uses a C# function

member (method) to calculate the distance between points. Note that "Point" class is computing the distance between two arbitrary points and the arbitrary values 1 & 2 as input parameters for the Stored Procedure "distance" whose definition is show below in **Figure 28 – Distance Stored Procedure.**

Figure 28 – Distance Stored Procedure

```
CREATE procedure clr..distance (in x1 integer, in y1 integer, in x2
integer, in y2 integer)
returns float
 {
 return new DB.DBA.Point(x1, y1).distance (new DB.DBA.Point (x2, y2));
}
```



Step 8. Once the distance stored procedure has been created, we then publish the procedure as a web service using the method previously shown in the section ***Publishing SQL Stored Procedures as Web Services.***

Step 9. Using the Virtuoso generated vsmx page, a SOAP Service can be invoked for the procedure for testing as seen **Figure 29 – Testing CLR UDT and distance stored procedure** and **Figure 30 - Testing the SOAP Operation, Distance**.

Figure 29 – Testing CLR UDT and distance stored procedure

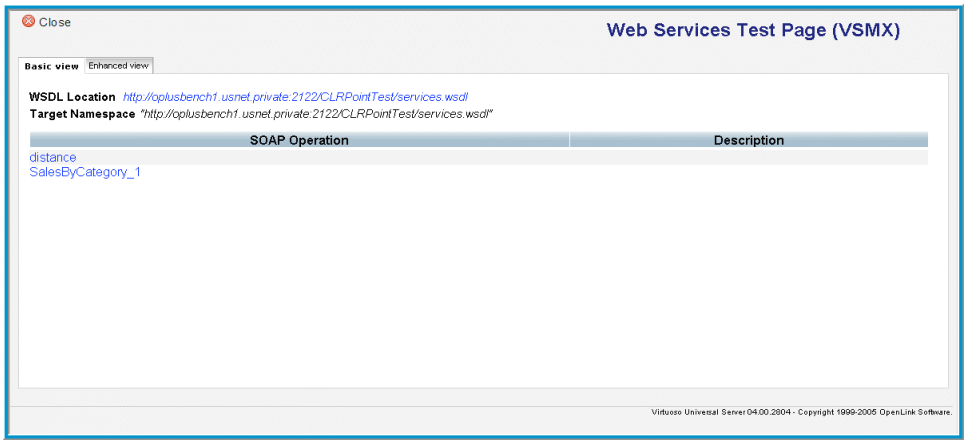


Figure 30 - Testing the SOAP Operation, Distance

3.3.2.1.2 *User Defined Types and Java Classes*

The steps used to create a hosted module in Java are the same as the CLR. However, in order to access the Java class instances they have to be defined as Virtuoso types using CREATE TYPE and specifying LANGUAGE JAVA. ***Figure 122*** and ***123*** shows the definition of the Point class written in Java and the UDT definition in Virtuoso.

Figure 31 Java code – Point.java

```
public class Point implements java.io.Serializable
{
  public double x = 0;
  public double y = 0;

  public Point (double new_x, double new_y)
    {
      x = new_x;
      y = new_y;
    }
  public double distance (Point p)
    {
      return Math.sqrt ((p.x - this.x) * (p.x - this.x) + (p.y - this.y) * (p.y -
this.y));
    }
```

```
    }
```

Figure 32 – Defining the Create Type in Virtuoso for Java

```
create type "Point" language JAVA external name 'Point'
 AS (
     "x" double precision external name 'x' external type 'D',
     "y" double precision external name 'y' external type 'D'
    )
 unrestricted
 CONSTRUCTOR METHOD "Point"  (
     "p1" double precision external type 'D',
     "p2" double precision external type 'D'),

 METHOD "distance" ("p1" "Point" external type 'LPoint;')
      returns double precision external type 'D'
      external name 'distance',
;
```

More Information

For complete details on User Defined Types, see:

- Virtuoso Documentation on User Defined Types
- Virtuoso CLR Using C# objects to extent Virtuoso via UDTs Tutorial
- Virtuoso Java Using Java objects to extent Virtuoso via UDTs Tutorial

3.3.2.2 .NET Assemblies

3.3.2.3 Scripting Languages

When a page in PHP, PERL or Python is in an executable directory in the web server space, no special code or declaration is required as in the aforementioned Java or CLR examples. These languages will be executed when requested. There is however special Virtuoso SQL specific functions for accessing these languages from stored procedures.

The following examples include PERL and Python code snippets followed by their SQL invocation from Virtuoso Interactive SQL (ISQL)***.***

3.3.2.3.1 *Perl Example*

In ***Figure 140***, the example shows direct PERL code execution from Virtuoso PL.

Figure 33 - Executing PERL directly from Virtuoso PL

```
select __hosting_http_handler ('pl', 'print "hello, perl"; ', vector (), vector (), 't1.pl');
```

Figures 141 and ***142*** show Perl file execution from Virtuoso PL.

Figure 34 – PERL Code test_print.pl

```
#!/usr/bin/perl
print "hello, perl from a file";
```

Figure 35 - Virtuoso PL CODE

```
select __hosting_http_handler ('pl', 'test_print.pl');
```

3.3.2.3.2 *Python Example:*

In ***Figure143***, the example shows direct python code execution from PL :

Figure 36 - Executing Python directly from Virtuoso PL

```
select __hosting_http_handler ('py', 'print "hello, python",; ', vector (), vector (), 't1.py');
```

Figures 144 and ***145*** demonstrate Python file execution from Virtuoso PL :

Figure 37 – Python Code test_print.py

```
#!/usr/bin/python
print "hello python from a file",;
```

Figure 38 – Virtuoso PL Code

```
select __hosting_http_handler ('py', 'test_print.py');
```

3.3.2.4 Creating an Assembly for .Net

In **Figure 39 – Create Library,** we have a create assembly example with the same class used in the CLR examples for .Net

Figure 39 – Create Library

```
CREATE LIBRARY "myPoint" as 'temp_dll_stor\\Point.dll')
WITH PERMISSION_SET = UNRESTRICTED
WITH AUTOREGISTER;
```

3.3.2.5 Sandboxes and Security

Hosted .Net code can run in two modes: Restricted and unrestricted.

3.3.2.5.1 *Restricted Mode*

In the restricted mode, the default, the code cannot take out of process connections, access the file system or perform any other possibly dangerous operations.

3.3.2.5.2 *Unrestricted Mode*

In the unrestricted mode, the .Net code will run with the privileges of a Windows account associated to the SQL account that is invoking the code. This defaults to the operating system account on behalf of which the Virtuoso server is running.

With Java, the code may be confined into a Java sandbox, as Java applets in browsers run or the code can run with the privileges of the OS account on which the Virtuoso server is running.

With both Java and .Net, the security setting is specified when importing the classes into Virtuoso.

3.3.3 Web Service Deployment

3.3.3.1 Virtuoso Hosted Services

3.3.3.2 3rd Party Web Services (e.g. asmx based services from Visual Studio)

Virtuoso provides the ability to host Microsoft .Net web services *.asmx* files without any programming. By inserting, an *.asmx* file in an executable virtual directory in either WebDAV or the file system will make its web services accessible to clients and will publish a WSDL file describing them. The mechanism used for exposing these files is similar to Virtuoso's ASP .Net hosting. This makes Virtuoso a viable alternative to Microsoft's IIS for .asmx hosting. Additionally, *.asmx* services can be hosted on Unix platforms through Virtuoso's integration of the Mono CLR run time providing alterative options for platform deployment for applications.

More Information

For complete details on setting up an environment to experiment with hosting and Writing ASP.Net Web Applications, see:

- Virtuoso Web Application Development Documentation

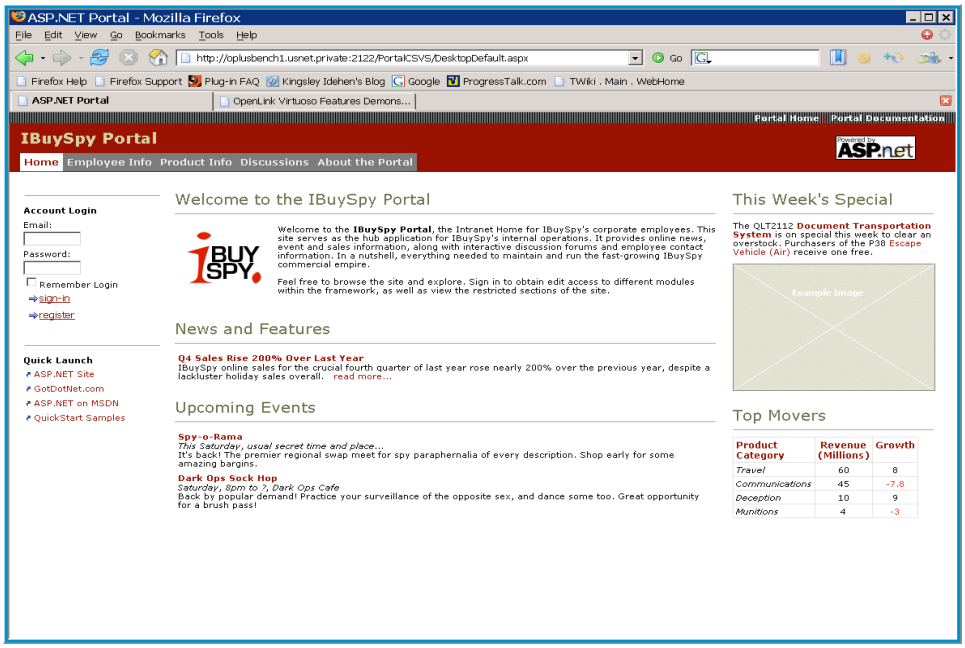
3.3.3.2.1 ASP .Net Hosting

The default the installed demo instance includes hosting of the IBuySpy Portal, which is an aspx-based application as seen below in **Figure 40 – IBuySpy Portal**

You can set this up through the tutorial or browse directly to this demonstration by typing the URL on your own local installation or on OpenLinks online demo:

http://localhost:8890/PortalCSVS/ on Windows or
http://demo.openlinksw.com:8890/PortalCS on Linux

Figure 40 – IBuySpy Portal



3.3.3.2.2 ASMX Hosting

Virtuoso can host ASMX files defining SOAP callable web services defined in any .Net language. The web service is deployed under Virtuoso simply by placing the ASMX file in either the DAV repository or a file system directory mapped to a Virtuoso web server virtual directory.

4 Appendix

4.1 Demo and Northwind Database

A number of the examples in this document reference the Virtuoso demo database, which is derived from Microsoft's Northwind database ships with Virtuoso 4.5. In addition, where indicated other examples specifically use the Northwind database. This database can be obtained from Microsoft at

http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en

4.2 Industry Standards Support

- **Runtimes & Frameworks**

  Microsoft .NET, Mono, J2EE

- **Web Services**

  SOAP, UDDI, WSDL, WS-Security, WS-Routing, WS-Reliable Messaging, WS-Policy, WS-Trust, BPEL4WS

- **XML**

  XPath, XQuery, XSL-T, XML Schema

- **Web & Internet**

  WebDAV, HTTP, SMTP, LDAP, POP3

- **SQL Data Access**

  SQL-200n, SQLX, ODBC, JDBC, ADO.NET, and OLE-DB.

4.3 Related Links

- Virtuoso On-line Tutorials and Demonstrations
  http://demo.openlinksw.com:/tutorial/
- Virutoso Blog - http://www.openlinksw.com/weblogs/virtuoso/.
- Documentation - http://docs.openlinksw.com/virtuoso/index.html
- For additional information about Virtuoso, see the OpenLink Virtuoso Product Web site . The resources available on this site include links to technical articles and white papers, downloads pricing, documentation, animated demonstrations and FAQs